test-meli-eda

April 9, 2024

[2]:
```
import pandas as pd
import datetime as dt
import numpy as np
import missingno as msno
import seaborn as sns
import matplotlib.pyplot as plt
```

Carrega e verifica o tipo de cada coluna no dataframe

[116]:
```
df = pd.read_csv('dados.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 20 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   a       150000 non-null  int64  
 1   b       137016 non-null  float64
 2   c       137016 non-null  float64
 3   d       149635 non-null  float64
 4   e       150000 non-null  float64
 5   f       149989 non-null  float64
 6   g       149806 non-null  object 
 7   h       150000 non-null  int64  
 8   i       150000 non-null  object 
 9   j       150000 non-null  object 
 10  k       150000 non-null  float64
 11  l       149989 non-null  float64
 12  m       149635 non-null  float64
 13  n       150000 non-null  int64  
 14  o       41143 non-null   object 
 15  p       150000 non-null  object 
 16  fecha   150000 non-null  object 
 17  monto   150000 non-null  float64
 18  score   150000 non-null  int64  
 19  fraude  150000 non-null  int64  
dtypes: float64(9), int64(5), object(6)
```

```
memory usage: 22.9+ MB
```

[117]:
```
df.head(15)
```

[117]:

	a	b	c	d	e	f	g	h	\
0	4	0.7685	94436.24	20.0	0.444828	1.0	BR	5	
1	4	0.7550	9258.50	1.0	0.000000	33.0	BR	0	
2	4	0.7455	242549.09	3.0	0.000000	19.0	AR	23	
3	4	0.7631	18923.90	50.0	0.482385	18.0	BR	23	
4	2	0.7315	5728.68	15.0	0.000000	1.0	BR	2	
5	4	0.7359	42727.15	50.0	0.000000	47.0	BR	1	
6	4	0.5962	7121.78	2.0	0.398000	0.0	BR	11	
7	4	0.6806	1656.95	50.0	1.043077	0.0	BR	11	
8	4	0.5893	311762.23	6.0	0.000000	15.0	AR	2	
9	4	0.4759	40143.12	50.0	0.000000	41.0	AR	3	
10	4	0.8353	121926.06	39.0	0.186957	34.0	BR	24	
11	4	0.7206	227874.02	1.0	0.230960	0.0	BR	8	
12	4	0.7688	440434.49	50.0	0.000000	78.0	BR	17	
13	4	0.7921	6555.99	50.0	0.447612	18.0	BR	9	
14	4	0.7702	706.47	1.0	0.153675	9.0	BR	7	

	i	j	k	\
0	Máquininha Corta Barba Cabelo Peito Perna Pelo…	cat_8d714cd	0.883598	
1	Avental Descartavel Manga Longa - 50 Un. Tnt …	cat_64b574b	0.376019	
2	Bicicleta Mountain Fire Bird Rodado 29 Alumini…	cat_e9110c5	0.516368	
3	Caneta Delineador Carimbo Olho Gatinho Longo 2…	cat_d06e653	0.154036	
4	Resident Evil Operation Raccoon City Ps3	cat_6c4cfdc	0.855798	
5	Kit Gamer Teclado Hedfone Mouse E Mousepad	cat_9d78e2e	0.571502	
6	Corpinho Avulso Joseph, Josepha Ou Placa Sem Sexo	cat_5d6059e	0.204991	
7	Tripa Para Fazer Linguiça - 45 Metros Long Short	cat_e686ce3	0.569230	
8	Soldadora Inverter 180 A + Máscara + 2 Esc. Ma…	cat_bfe5d9b	0.897001	
9	Gamepad Joystick Para Telefono Celular Android…	cat_5d79fb9	0.735790	
10	Escova Dental Curaprox 5460 Ultra Soft Com 3 U…	cat_4744ece	0.943792	
11	Par Lanterna Luz De Placa C/ Led Fiat 500 Cone…	cat_768556e	0.530758	
12	Câmera Sony Mirrorless Alpha A6500 (corpo) Gar…	cat_d3aa8de	0.618222	
13	Filtro Ar Motor Audi A3 Vw Golf Mk7 Após 2012 …	cat_ca18469	0.132371	
14	Base Alta Cobertura Nº4 Bege Essenze Di Pozzi	cat_708c94b	0.269991	

	l	m	n	o	p	fecha	monto	score	fraude
0	240.0	102.0	1	NaN	N	2020-03-27 11:51:16	5.64	66	0
1	4008.0	0.0	1	Y	N	2020-04-15 19:58:08	124.71	72	0
2	1779.0	77.0	1	NaN	N	2020-03-25 18:13:38	339.32	95	0
3	1704.0	1147.0	1	NaN	Y	2020-04-16 16:03:10	3.54	2	0
4	1025.0	150.0	1	NaN	N	2020-04-02 10:24:45	3.53	76	0
5	2798.0	506.0	1	NaN	Y	2020-04-13 18:42:28	28.00	32	0
6	127.0	125.0	0	NaN	N	2020-03-22 19:20:24	10.56	71	0
7	363.0	224.0	0	NaN	N	2020-04-12 11:49:54	6.13	81	0

8	4661.0	826.0	1	Y	Y	2020-04-20 22:58:52	142.71	64	0
9	4701.0	940.0	1	NaN	Y	2020-03-11 15:06:38	30.48	60	0
10	3248.0	132.0	1	Y	Y	2020-03-15 19:18:50	14.85	10	0
11	2640.0	0.0	1	NaN	N	2020-04-19 17:07:37	23.79	24	0
12	4970.0	1580.0	1	NaN	Y	2020-03-16 11:49:05	1138.64	83	0
13	5513.0	1060.0	1	Y	Y	2020-04-02 22:30:29	7.89	50	0
14	3199.0	0.0	1	NaN	N	2020-03-20 15:42:14	9.19	60	0

Transforma a coluna "fecha" para o formato datetime. Cria a coluna hora que indica a hora da compra.

[118]:
```
df = df.assign(fecha = df.fecha.apply(lambda x:dt.datetime.strptime(x,␣
  ↪"%Y-%m-%d %H:%M:%S")))
df = df.assign(hora = df.fecha.dt.hour+df.fecha.dt.minute/60)
```

Aqui vamos criar duas listas para identificar colunas categóricas e numéricas.

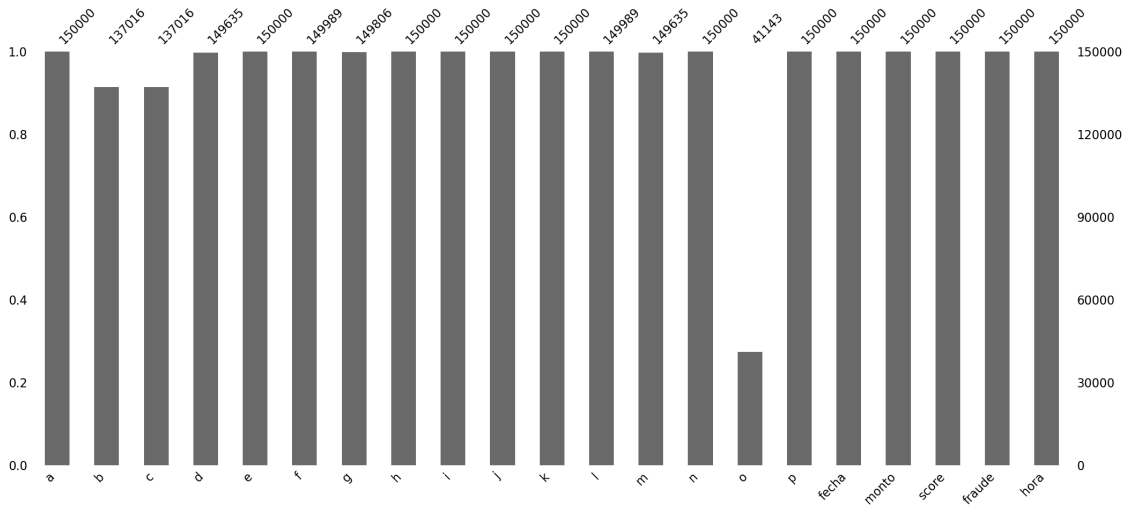
[6]:
```
cat_columns = ['a', 'g', 'i', 'j', 'o', 'n', 'p']
num_cols = ['b', 'c', 'd', 'e', 'f', 'h', 'k', 'l', 'm', 'monto', 'hora']
```

0.0.1 Análise de Dados faltantes

Dentre todas as colunas, somente as colunas **b**,**c** e **o** contém dados faltantes.

[7]:
```
msno.bar(df)
```

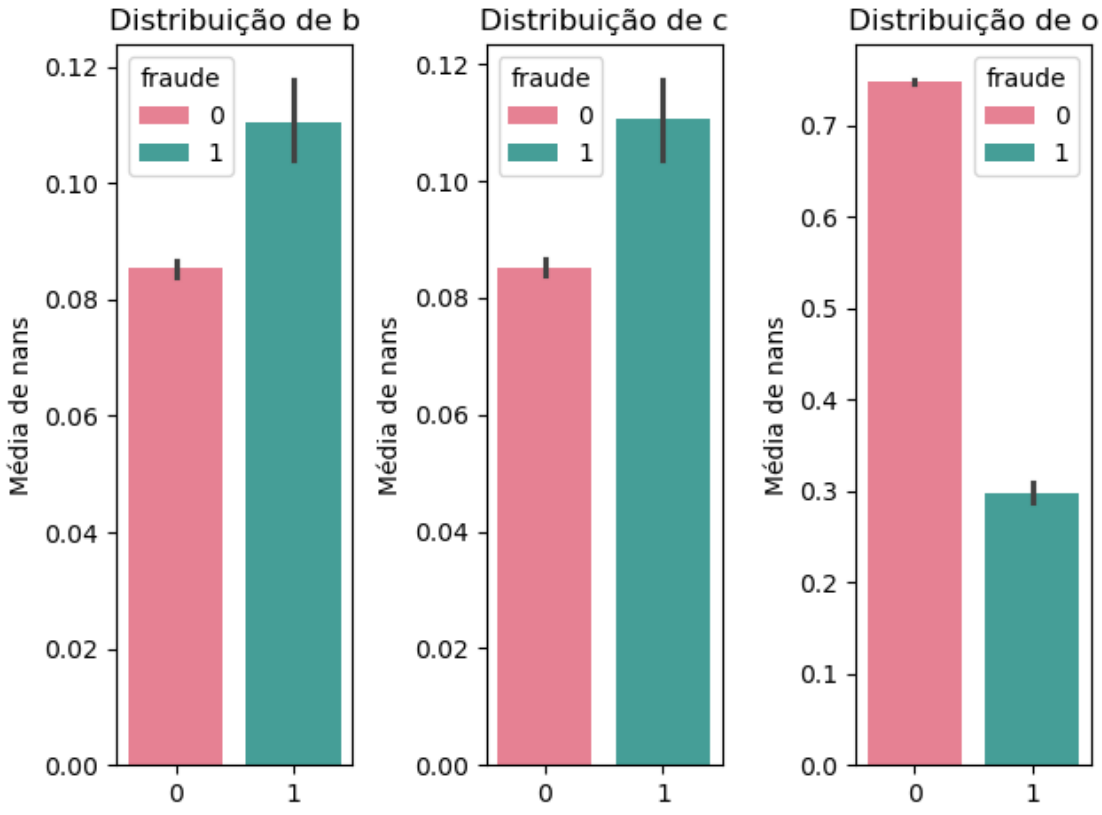
[7]: <Axes: >



A proporção de dados faltantes pode ser um fator discriminativo. De fato, observando o plot abaixo, verificamos que a proporção de dados faltantes em **b** e em **c** para fraudes é maior. O

inverso acontece com **o**, onde a média de dados faltantes é menos da metade quando o pagamento é uma fraude.

[75]:
```
cols_with_nan = ['b','c', 'o']
#if nan replace with 1 otherwise 0
df_nan = pd.DataFrame(np.where(df[cols_with_nan].isnull(),1,0),␣
 ↪columns=cols_with_nan)
df_nan = df_nan.assign(fraude = df['fraude'])

for col in cols_with_nan:
    plt.subplot(1, 3, cols_with_nan.index(col) + 1)
    sns.barplot(df_nan, x="fraude", y=col, hue="fraude", palette='husl')
    plt.xlabel('')
    plt.title(f'Distribuição de {col}')
    plt.ylabel('Média de nans')

plt.tight_layout()
plt.show()
```

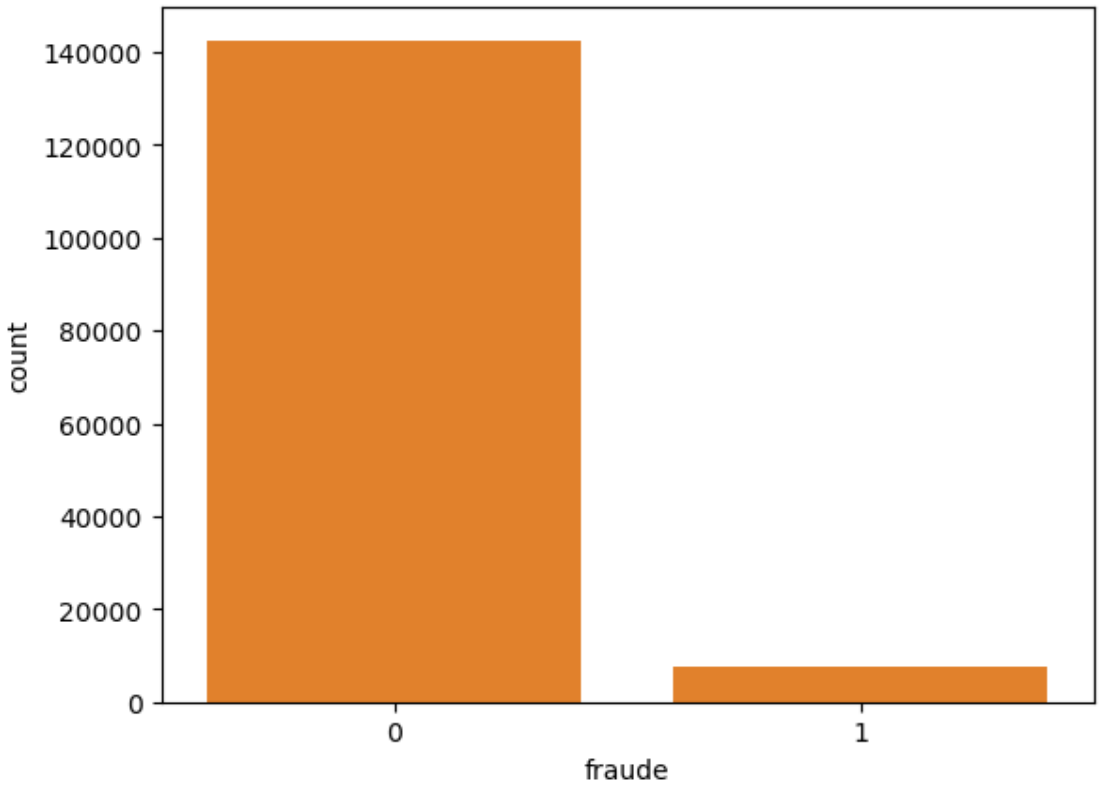


0.0.2 Proporção de fraudes

É comum que as bases de treinamento de fraudes exibam desequilíbrio, o que pode afetar a eficácia dos modelos de detecção. Nesse contexto, diversos métodos de oversampling e undersampling podem ser empregados para lidar com esse desafio, buscando garantir uma representação adequada das classes. Considerando que uma proporção de 5% não é considerada prejudicial, podemos adotar estratégias para ajustar os pesos dos modelos, visando mitigar o impacto do desbalanceamento e melhorar a capacidade de identificação de fraudes de forma mais precisa e eficiente.

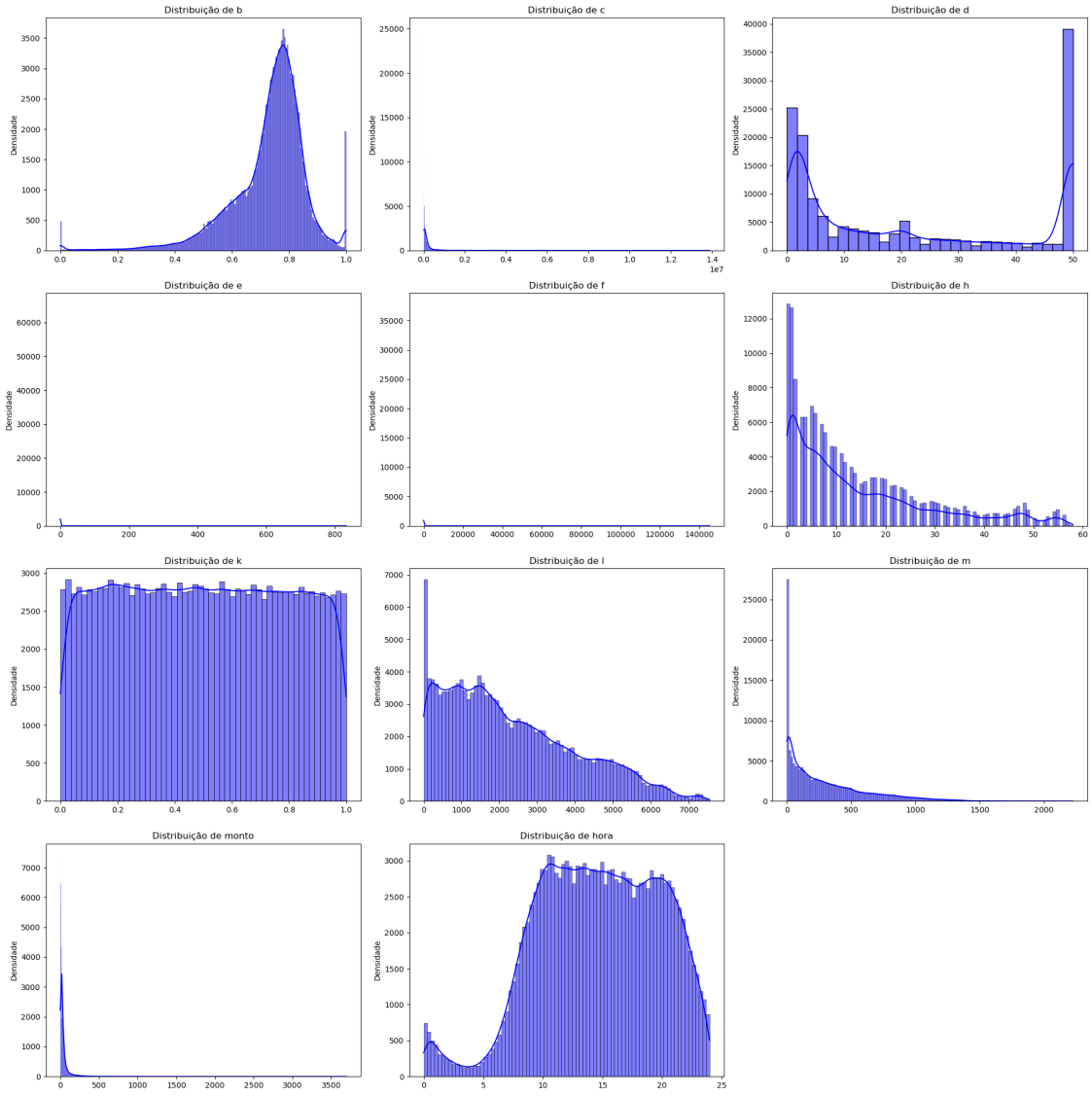
[9]:
```
print('label 0:', round(
        df['fraude'].value_counts()[0]/len(df)*100, 2), '% of datapoints')
print('label 1:', round(
        df['fraude'].value_counts()[1]/len(df)*100, 2), '% of datapoints')
sns.countplot(df, x="fraude")

sns.countplot(df, x="fraude");
```

```
label 0: 95.0 % of datapoints
label 1: 5.0 % of datapoints
```



0.0.3 Distribuição das colunas numéricas

[10]:
```
plt.figure(figsize=(20, 20))
for col in num_cols:
    plt.subplot(4, 3, num_cols.index(col) + 1, alpha=0.7)
    sns.histplot(df[col], kde=True, color='blue')
    plt.title(f'Distribuição de {col}')
    plt.xlabel('')
    plt.ylabel('Densidade')

plt.tight_layout()
plt.show();
```

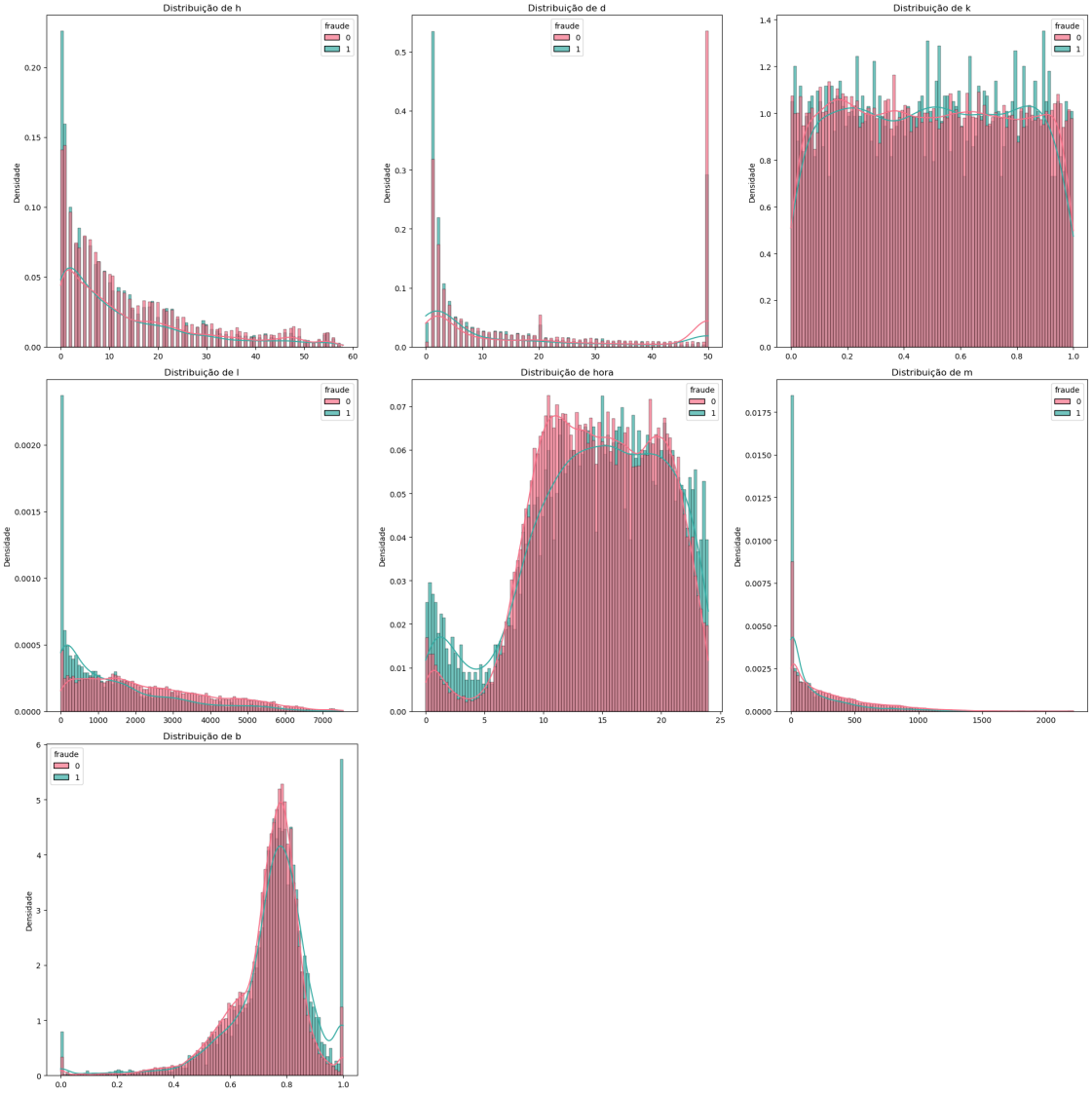


Para algumas colunas, a visualização não está clara. Portanto, vamos aplicar a escala logarítmica

para melhorar a compreensão.

[11]:
```
log_nums_cols = ['c', 'e', 'f', 'monto']
```

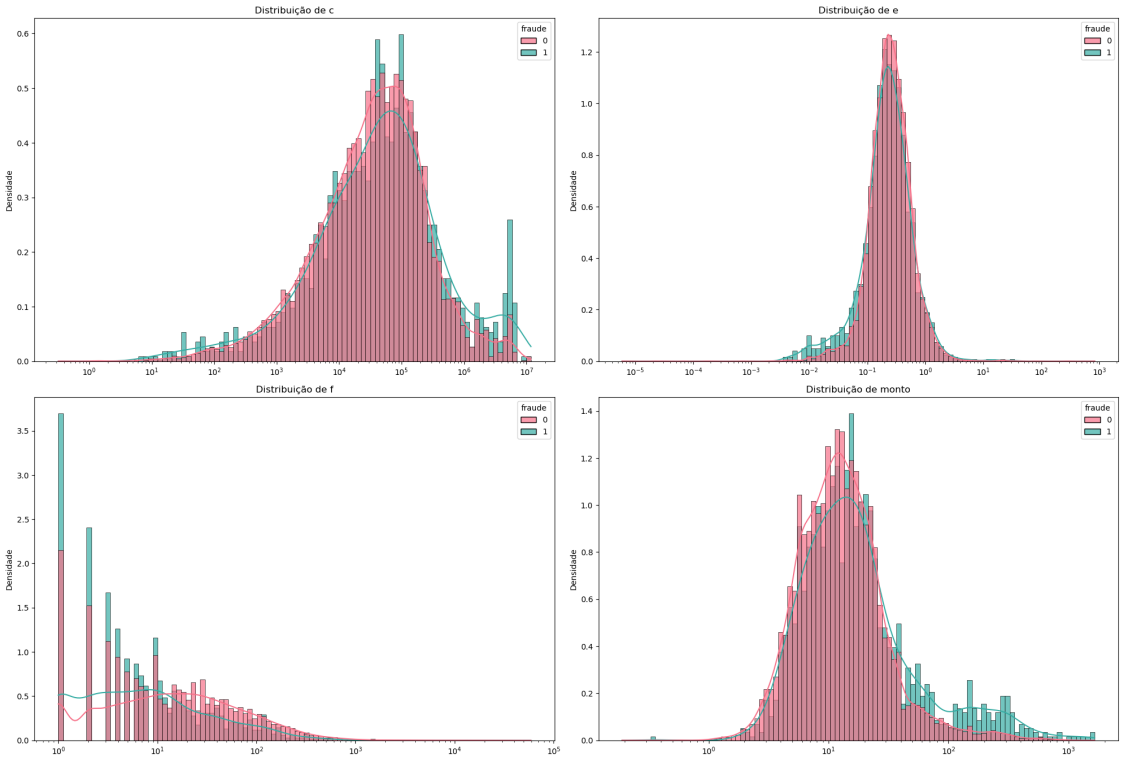
[13]:
```
plt.figure(figsize=(20, 20))
cols = set(num_cols)-set(log_nums_cols)
for col in cols:
    plt.subplot(3, 3, list(cols).index(col) + 1)
    sns.histplot(data=df.dropna(), x=col, hue='fraude', kde=True, bins=100,␣
 ↪alpha=0.7, palette='husl', stat='density', common_norm=False)
    plt.title(f'Distribuição de {col}')
    plt.xlabel('')
    plt.ylabel('Densidade')

plt.tight_layout()
plt.show()
```



Comparar a densidade da distribuição para o caso de fraude e não-fraudes nos ajudar a entender algumas diferenças de comportamento: - **Hora**: As probabilidades de fraude aumentam durante a noite; - **l**, **h**, **d** e **m**: Com valores mais baixos, as chances de ocorrência de fraudes aumentam; - **b**: As probabilidades de fraude aumentam quando o valor é igual a 1.

Distribuições log-normais

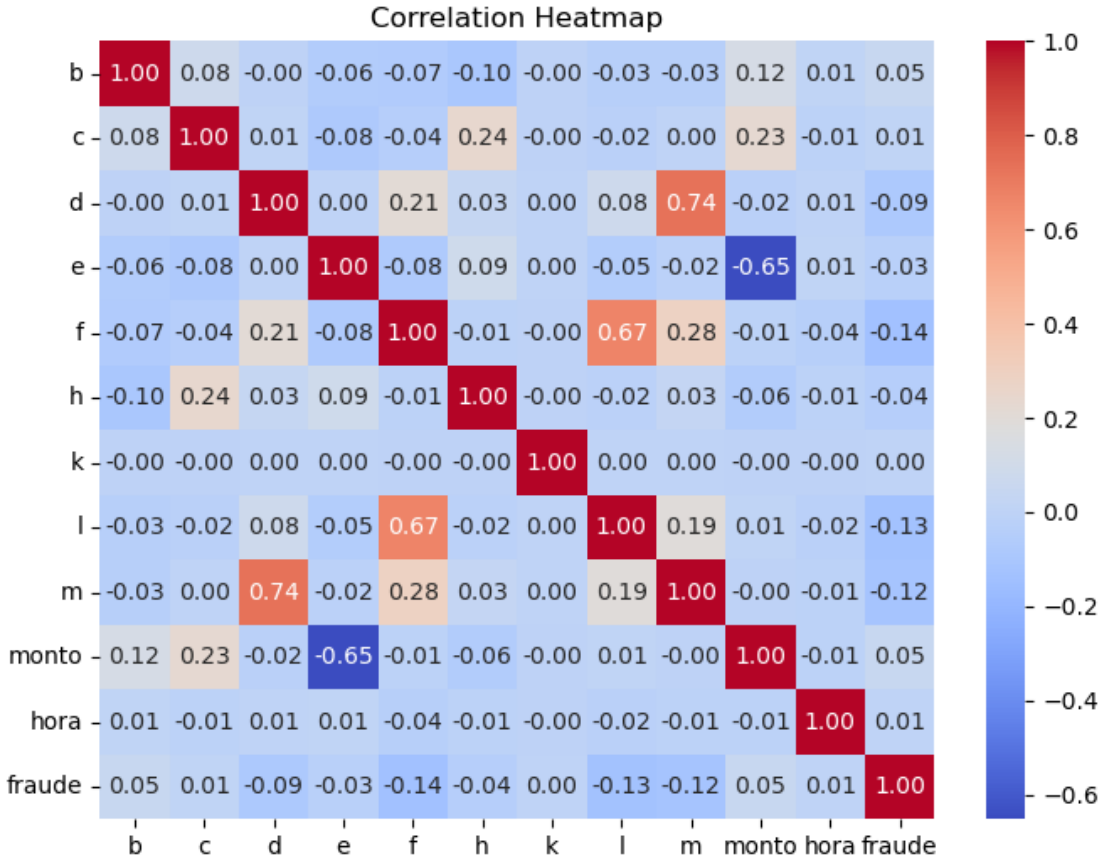
[14]:
```
plt.figure(figsize=(20, 20))

# Fazendo o replace nas colunas especificadas
df_temp = df.replace({col: {0: pd.NA} for col in log_nums_cols}).dropna()
for col in log_nums_cols:
    plt.subplot(3, 2, log_nums_cols.index(col) + 1)
    sns.histplot(data=df_temp, x=col, hue='fraude', kde=True, bins=100, alpha=0.
 ↪7, palette='husl', stat='density', common_norm=False, log_scale=True)
    plt.title(f'Distribuição de {col}')
    plt.xlabel('')
    plt.ylabel('Densidade')

plt.tight_layout()
plt.show()
```



- **monto** - As chances de fraude aumentam com valores mais altos;
- **f** - As chances de fraude aumentam com valores mais baixos.

0.0.4 Estudo de Correlação

[56]:
```
plt.figure(figsize=(8, 6))
df_corr = df[num_cols+['fraude']].corr('spearman')
sns.heatmap(df_corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



[74]:
```
muito_forte = 0.8
forte = 0.6
moderada = 0.4
fraca = 0.20

df_corr_upper = df_corr.where(np.triu(np.ones(df_corr.shape), k=1).astype(bool))
correlacoes = df_corr_upper.unstack().sort_values(ascending=False)

# Filtra as correlações muito fortes
```

```
print("Correlações muito fortes (maior que 0.8):")
print(correlacoes[(correlacoes != 1) & (correlacoes >= muito_forte)])

# Filtra as correlações fortes
print("\nCorrelações fortes (entre 0.6 e 0.79 ):")
print(correlacoes[(correlacoes != 1) & (correlacoes >= forte) & (correlacoes <␣
 ↪muito_forte)])

# Filtra as correlações moderadas
print("\nCorrelações moderadas (entre  0.4 e 0.59):")
print(correlacoes[(correlacoes != 1) & (correlacoes >= moderada) & (correlacoes␣
 ↪< forte)])

# Filtra as correlações fracas
print("\nCorrelações fracas  (entre 0.20 e 0.39:")
print(correlacoes[(correlacoes != 1) & (correlacoes >= fraca) & (correlacoes <␣
 ↪moderada)])
```

```
Correlações muito fortes (maior que 0.8):
Series([], dtype: float64)

Correlações fortes (entre 0.6 e 0.79 ):
m  d    0.735300
l  f    0.669363
dtype: float64

Correlações moderadas (entre  0.4 e 0.59):
Series([], dtype: float64)

Correlações fracas  (entre 0.20 e 0.39:
m      f    0.284475
h      c    0.242640
monto  c    0.230002
f      d    0.210739
dtype: float64
```

A análise de correlação é útil para eliminar variáveis redundantes ou irrelevantes do conjunto de dados, o que pode melhorar o desempenho do modelo e reduzir a complexidade computacional. No entanto, é importante observar que a correlação não implica causalidade. Mesmo que duas variáveis estejam fortemente correlacionadas, isso não significa necessariamente que uma causa a outra. A correlação apenas descreve a relação estatística entre as variáveis, e é necessário realizar análises adicionais para determinar a relação causal, se houver.

0.0.5 Análise Temporal

[17]:
```
df['data_dia'] = df['fecha'].dt.strftime('%Y-%m-%d')
print("Essa base contém ", len(df), "pagamentos")
print("A data mais antiga é", min(df["data_dia"]))
print("A data mais recente é", max(df["data_dia"]))
```

```
Essa base contém  150000 pagamentos
A data mais antiga é 2020-03-08
A data mais recente é 2020-04-21
```

[18]:
```
df_conversao = df.groupby(["data_dia", "fraude"]).agg({"monto": 'sum'}).
 ↪reset_index()
df_conversao
```

[18]:

	data_dia	fraude	monto
0	2020-03-08	0	113830.82
1	2020-03-08	1	9798.05
2	2020-03-09	0	190010.34
3	2020-03-09	1	14666.02
4	2020-03-10	0	182384.29
..	…	…	…
85	2020-04-19	1	9028.48
86	2020-04-20	0	208400.31
87	2020-04-20	1	23163.98
88	2020-04-21	0	193130.16
89	2020-04-21	1	17321.24

[90 rows x 3 columns]

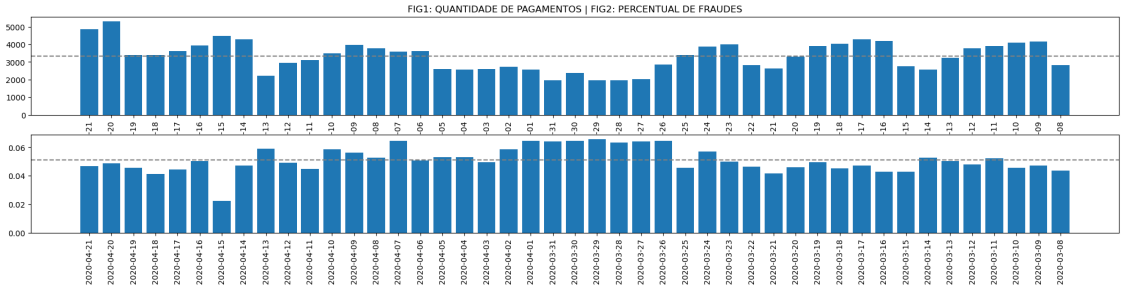
[19]:
```
import plotly.graph_objs as go

soma_fraudes = df[df['fraude'] == 1].groupby('data_dia')['monto'].sum()
soma_nao_fraudes = df[df['fraude'] == 0].groupby('data_dia')['monto'].sum()

# Plotando o gráfico com Plotly
fig = go.Figure()

fig.add_trace(go.Bar(x=soma_fraudes.index, y=soma_fraudes, name='Fraudes'))
fig.add_trace(go.Bar(x=soma_nao_fraudes.index, y=soma_nao_fraudes, name='Não␣
 ↪Fraudes'))

fig.update_layout(title='Soma do Montante por Dia',
                  xaxis_title='Data Dia',
                  yaxis_title='Soma',
                  barmode='stack')
```

```
fig.show()
```

[20]:
```
import plotly.express as px
per_monto_fraudes= soma_fraudes/soma_nao_fraudes

fig = px.line(per_monto_fraudes, x=per_monto_fraudes.index, y='monto',␣
 ↪title='Percentual do montante que são fraudes')
fig.update_xaxes(title='Dia')
fig.update_yaxes(title='Porcentagem')
fig.show()
```

Durante a análise do gráfico que representa o percentual de fraudes por dia, foi observado que em determinados períodos temporais (21-03, 03-04, 15-04), a incidência de fraudes foi significativamente reduzida, denotando um baixo volume de ocorrências fraudulentas. Em contrapartida, em outras janelas temporais, verificou-se uma elevação considerável no número de fraudes registradas (01-04 por exemplo), indicando a presença de períodos com uma incidência substancialmente maior de atividades fraudulentas. Esta variabilidade nos níveis de fraudes ao longo do tempo sugere a influência de fatores dinâmicos ou sazonais que podem impactar a segurança e a integridade do sistema ou processo em análise.

[33]:
```
df_conversao_dia = df.groupby("data_dia").agg({"j": 'count', "fraude": 'sum'}).
 ↪reset_index()
df_conversao_dia["conversao_per"] = df_conversao_dia["fraude"]/
 ↪df_conversao_dia["j"]
df_conversao_dia=df_conversao_dia.sort_values("data_dia", ascending = False).
 ↪rename(columns={'j': 'n_pagamentos', 'fraude': 'n_fraudes'})

plt.figure(figsize=(25, 5))

# QUANTIDADE DE VENDAS POR PRAÇA:
plt.subplot(211)
df_conversao_dia=df_conversao_dia.sort_values("data_dia", ascending = False)
plt.bar(df_conversao_dia["data_dia"], df_conversao_dia["n_pagamentos"])
plt.axhline(y=df_conversao_dia.n_pagamentos.mean(), color='gray',␣
 ↪linestyle='--')
plt.title("FIG1: QUANTIDADE DE PAGAMENTOS | FIG2: PERCENTUAL DE FRAUDES")
plt.xticks(rotation=90)

# % CONVERSÃO POR PRAÇA:
plt.subplot(212)
plt.xticks(rotation=90)
plt.bar(df_conversao_dia["data_dia"], df_conversao_dia["conversao_per"])
plt.axhline(y=df_conversao_dia.conversao_per.mean(), color='gray',␣
 ↪linestyle='--');
```



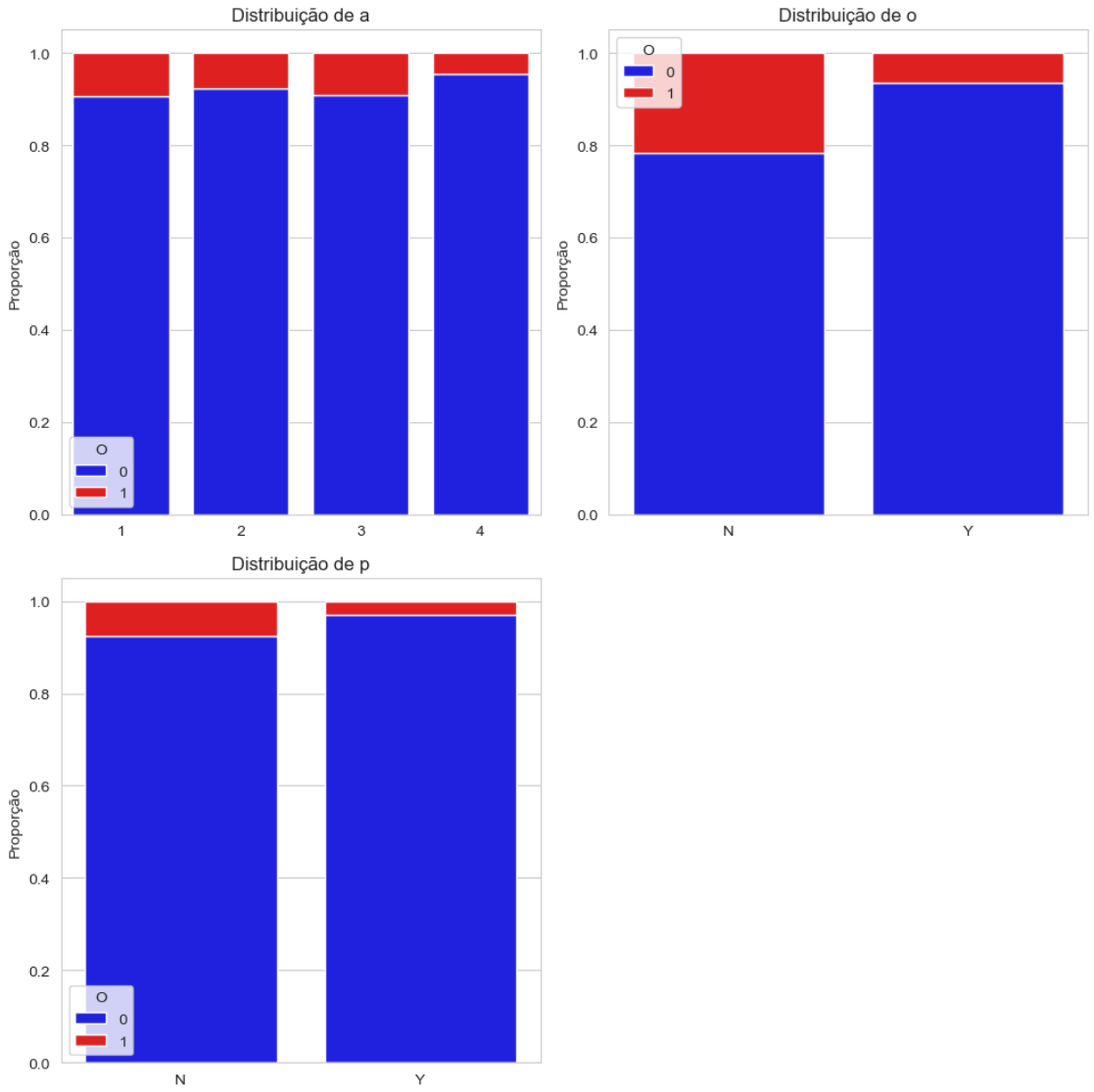
[36]:
```
from scipy.stats import pearsonr
pearsonr(df_conversao_dia['n_pagamentos'], df_conversao_dia['conversao_per'])
```

[36]:
```
PearsonRResult(statistic=-0.5349408710592065, pvalue=0.0001532036208505364)
```

Com significância estatística (p-value<0.05) pode-se dizer que há uma correlação inversa moderada entre o número de pagamentos e o percentual de fraudes por dia. Uma possível explicação é que, quando ocorre um aumento significativo no número de pagamentos, o número de fraudes não crescem na mesma proporção, diminuindo o percentual.

0.0.6 Distribuição das colunas categóricas

[133]:
```
plt.figure(figsize=(10, 10))

# Fazendo o replace nas colunas especificadas
for i, col in enumerate(['a', 'o', 'p'], start=1):
    df_temp = df.groupby(col)['fraude'].value_counts(normalize=True).unstack().
 ↪reset_index()
    df_temp['fraude'] = df_temp.index
    plt.subplot(2, 2, i)
    sns.barplot(data=df_temp, x=col, y=0, color='b', label='0')
    sns.barplot(data=df_temp, x=col, y=1, color='r', label='1',␣
 ↪bottom=df_temp[0])
    plt.title(f'Distribuição de {col}')
    plt.xlabel('')
    plt.ylabel('Proporção')
    plt.legend(title='0')

plt.tight_layout()
plt.show()
```



- **a** - Menor proporçao de fraudes para valor igual a 4, em comparação com outros valores;
- **o** e **p** - Menor proporção de fraudes para Y em em comparação com N.

Para uma análise mais completa e para validar a significância estatística das conclusões sobre a diferença de proporções, é comum realizar um teste de hipóteses, como o teste de diferença de proporções z.

0.0.7 Fraudes por Região

[312]:
```
df_conversao_regiao = df.groupby("g").agg({"j": 'count', "fraude": 'sum'}).
  ↪reset_index()
df_conversao_regiao["conversao_per"] = df_conversao_regiao["fraude"]/
  ↪df_conversao_regiao["j"]
```

```
df_conversao_regiao=df_conversao_regiao.sort_values("fraude", ascending =␣
 ↪False).rename(columns={'j': 'n_pagamentos', 'fraude': 'n_fraudes'})

plt.figure(figsize=(25, 5))

# QUANTIDADE DE VENDAS POR PRAÇA:
plt.subplot(211)
df_conversao_regiao=df_conversao_regiao.sort_values("n_fraudes", ascending =␣
 ↪False)
plt.bar(df_conversao_regiao["g"], df_conversao_regiao["n_fraudes"])
plt.title("FIG1: QUANTIDADE DE PAGAMENTOS | FIG2: PERCENTUAL DE FRAUDES")
plt.xticks(rotation=90)

# % CONVERSÃO POR PRAÇA:
plt.subplot(212)
plt.xticks(rotation=90)
plt.bar(df_conversao_regiao["g"], df_conversao_regiao["conversao_per"]);
```

Algumas regiões apresentam uma taxa de ocorrência de fraudes consideravelmente elevada, enquanto outras não registram nenhuma incidência de fraude. No entanto, é imprescindível realizar uma avaliação da significância estatística dessa proporção de fraudes por região, visto que muitas delas possuem um baixo volume de transações ou pagamentos, o que pode influenciar na interpretação dos resultados.

[54]:
```
from scipy.stats import binomtest

p_values = []

for idx, row in df_conversao_regiao.iterrows():
    n_sucesso = row['n_fraudes']
    n_amostra = row['n_pagamentos']

    p_value = binomtest(n_sucesso, n_amostra, p=0.5,  alternative='two-sided').
 ↪pvalue

    # Adicionando o p-value à lista
    p_values.append(p_value)
```

```
df_conversao_regiao['p_value'] = p_values
df_conversao_regiao['is_significative'] = df_conversao_regiao['p_value'] < 0.
↪0001

# Exibindo o DataFrame com o p-value para cada país
df_conversao_regiao
```

[54]:

	g	n_pagamentos	n_fraudes	conversao_per	p_value	is_significative
6	BR	111628	6162	0.055201	0.000000e+00	True
2	AR	31964	1179	0.036885	0.000000e+00	True
48	US	2273	70	0.030796	0.000000e+00	True
49	UY	2967	29	0.009774	0.000000e+00	True
44	SE	358	15	0.041899	4.113443e-82	True
43	RU	73	6	0.082192	3.947261e-14	True
18	ES	69	5	0.072464	4.119231e-14	True
32	MX	236	3	0.012712	3.968030e-65	True
20	FR	18	3	0.166667	7.537842e-03	False
14	DE	9	3	0.333333	5.078125e-01	False
34	NL	2	2	1.000000	5.000000e-01	False
16	EC	4	2	0.500000	1.000000e+00	False
30	KR	1	1	1.000000	1.000000e+00	False
19	FI	1	1	1.000000	1.000000e+00	False
36	NZ	3	1	0.333333	1.000000e+00	False
41	PT	13	1	0.076923	3.417969e-03	False
12	CO	64	1	0.015625	7.047314e-18	True
10	CL	9	1	0.111111	3.906250e-02	False
35	NO	1	0	0.000000	1.000000e+00	False
38	PE	5	0	0.000000	6.250000e-02	False
33	NI	1	0	0.000000	1.000000e+00	False
37	PA	3	0	0.000000	2.500000e-01	False
0	AD	1	0	0.000000	1.000000e+00	False
39	PH	1	0	0.000000	1.000000e+00	False
40	PR	2	0	0.000000	5.000000e-01	False
42	PY	4	0	0.000000	1.250000e-01	False
45	TR	1	0	0.000000	1.000000e+00	False
46	TW	1	0	0.000000	1.000000e+00	False
47	UA	3	0	0.000000	2.500000e-01	False
31	LB	2	0	0.000000	5.000000e-01	False
25	HN	1	0	0.000000	1.000000e+00	False
29	JP	1	0	0.000000	1.000000e+00	False
28	IT	14	0	0.000000	1.220703e-04	False
3	AU	3	0	0.000000	2.500000e-01	False
4	BE	3	0	0.000000	2.500000e-01	False
5	BO	2	0	0.000000	5.000000e-01	False
7	BS	1	0	0.000000	1.000000e+00	False
8	CA	3	0	0.000000	2.500000e-01	False

9	CH	3	0	0.000000	2.500000e-01	False
11	CN	3	0	0.000000	2.500000e-01	False
13	CR	2	0	0.000000	5.000000e-01	False
15	DO	1	0	0.000000	1.000000e+00	False
17	EG	1	0	0.000000	1.000000e+00	False
21	GB	43	0	0.000000	2.273737e-13	True
22	GE	1	0	0.000000	1.000000e+00	False
23	GH	1	0	0.000000	1.000000e+00	False
24	GR	1	0	0.000000	1.000000e+00	False
1	AE	2	0	0.000000	5.000000e-01	False
26	IL	1	0	0.000000	1.000000e+00	False
27	IN	1	0	0.000000	1.000000e+00	False
50	ZA	1	0	0.000000	1.000000e+00	False

Com base nesta tabela, é possível observar que alguns países apresentam uma significância estatística em relação ao número de fraudes (considerando p-value<0.0001), enquanto outros não. Isso significa que, para esses países, o número de fraudes identificado não é atribuível ao acaso, e podemos assegurar que a proporção indicada é a propoção esperada de fraudes para estes países.

Com base nessas informações, temos o gráfico revisado:

[55]:
```
df_conversao_regiao_new = df_conversao_regiao[df_conversao_regiao.
 ↪is_significative]

plt.figure(figsize=(25, 5))

# QUANTIDADE DE VENDAS POR PRAÇA:
plt.subplot(211)
df_conversao_regiao_new=df_conversao_regiao_new.sort_values("conversao_per",␣
 ↪ascending = False,)
plt.bar(df_conversao_regiao_new["g"], df_conversao_regiao_new["n_fraudes"])
plt.title("FIG1: QUANTIDADE DE PAGAMENTOS | FIG2: PERCENTUAL DE FRAUDES")
plt.xticks(rotation=90)

# % CONVERSÃO POR PRAÇA:
plt.subplot(212)
plt.xticks(rotation=90)
plt.bar(df_conversao_regiao_new["g"], df_conversao_regiao_new["conversao_per"]);
```